# 服务器

## 防火墙端口

添加端口：firewall-cmd --zone=public --add-port=80/tcp --permanent

firewall-cmd --reload

查看端口状态：firewall-cmd --query-port=9200/tcp

## Nginx

查看配置文件位置 nginx –t

配置

1. upstream cluster1 {
2. server localhost:8080;
3. server localhost:8081;
4. }
6. server {
7. listen       81;
8. server\_name  localhost;
9. client\_max\_body\_size 1024M;
11. location /home {
12. proxy\_pass http://cluster1;
13. proxy\_set\_header Host $host:$server\_port;
14. }
15. }

上面upstream默认负载均衡策略为RR

http服务器

1. # 所有静态请求都由nginx处理，存放目录为html
2. location ~ \.(gif|jpg|jpeg|png|bmp|swf|css|js)${
3. root    e:\wwwroot;
4. }
6. # 所有动态请求都转发给tomcat处理
7. location ~\.(jsp|do)$ {
8. proxy\_pass  http://test;
9. }
10. error\_page  500 502 503 504 /50x.html;
11. location =/50x.html {
12. root   e:\wwwroot;
13. }

Configuration summary

+ using system PCRE library

+ OpenSSL library is not used

+ using system zlib library

nginx path prefix: "/usr/local/nginx"

nginx binary file: "/usr/local/nginx/sbin/nginx"

nginx modules path: "/usr/local/nginx/modules"

nginx configuration prefix: "/usr/local/nginx/conf"

nginx configuration file: "/usr/local/nginx/conf/nginx.conf"

nginx pid file: "/usr/local/nginx/logs/nginx.pid"

nginx error log file: "/usr/local/nginx/logs/error.log"

nginx http access log file: "/usr/local/nginx/logs/access.log"

nginx http client request body temporary files: "client\_body\_temp"

nginx http proxy temporary files: "proxy\_temp"

nginx http fastcgi temporary files: "fastcgi\_temp"

nginx http uwsgi temporary files: "uwsgi\_temp"

nginx http scgi temporary files: "scgi\_temp"

1)启动操作

nginx -c /usr/local/nginx/conf/nginx.conf

2)停止操作

从容停止Nginx

快速停止Nginx：

kill -TERM 主进程号

强制停止Nginx：

kill -9 主进程号

3)

nginx -s reload

reload

./nginx -s reload

reload会重新加载配置文件，服务不中断，出错会rollback用上一次正确配置文件。

Request\_body

[Reference](http://www.madblog.cn/posts/e4dd5bbae50fa621.html)

## Mysql

Mgr

mysql [-h slc11fsp.us.oracle.com] -uroot -p

启动服务

service mysqld start

版本：mysql –V；或者select version();

Tables：show databases/tables;

查看表结构

desc table\_name;

Show DDl: show create table ld\_cart;

indexs

Show: show index from fund\_history\G;

Create: create index idx\_fund\_name on fund(name);

GRANT ALL PRIVILEGES ON \*.\* TO 'root'@'%' IDENTIFIED BY '123456' WITH GRANT OPTION;

## Linux

[Reference](https://www.linuxidc.com/Linux/2014-10/108014.htm)

进入安装界面，按e，修改参数

linuxefi /images/pxeboot/vmlinuz inst.stage2=hd:/dev/sdb4 nomodeset quiet

initrdefi /images/pxeboot/initrd.img

然后使用Ctrl+X启动即可进入正常安装流程

# 大数据

## Redis

redis

1)启动： redis-server.exe redis.windows.conf

开启memcache，并连接测试：

以守护进程模式启动memcache

memcached-1.5.2/bin/memcached -d -l slc11fsp.us.oracle.com -p 11211 -m 2048 -u root

## Flume

启动

bin/flume-ng agent -c ./conf -f ./conf/2.conf -n agent1 -Dflume.root.logger=INFO,console

* -c conf 指定flume自身的配置文件所在目录
* -f conf/netcat-logger.con 指定我们所描述的采集方案
* -n a1 指定agent的名字

组合

[Reference](http://www.voidcn.com/article/p-wgqahebc-ev.html)

## Es

安装 [Reference](https://www.cnblogs.com/yufeng218/p/8331438.html)

启动es

su elastic

/usr/local/elasticsearch-5.6.0/bin/elasticsearch –d

chown elastic /usr/local/elasticsearch-6.8.0/ -R

外网访问

vi elasticsearch.yml

network.bind\_host: 0.0.0.0 (network.host: 0.0.0.0 5.6?)

安装问题汇总

<https://blog.51cto.com/10950710/2124131>

vm

sysctl -w vm.max\_map\_count=262144

修改jvm ops

vi config/jvm.options

3.修改es文件拥有着 chown elastic:elastic -R ./\*

### kibana

1.kibana.yml server.host: "0.0.0.0"

Test url

http://localhost:5601/

## Kafka

### 安装

[Reference](https://juejin.im/post/5c1f9b3bf265da61590ba9b1)

ver: kafka\_2.11-2.1.0

启动/停止

* 启动 ./bin/kafka-server-start.sh ./config/server.properties &
* 停止 ./bin/kafka-server-stop.sh ./config/server.properties &

### 配置

vi ./config/server.properties

1.Zookeeper

* zookeeper.connect=localhost:2181
* zookeeper.connection.timeout.ms=6000

2. socket settings

* listeners：the socket server listens on
* advertised.listeners：broker的地址。配置外网访问.(0.9.x后新增，废弃了 advertised.host.name 和 advertised.host.port)

3.broker

broker.id：broker id

### Topic

* Topic列表： bin/kafka-topics.sh -list -zookeeper localhost:2181
* 创建topic：bin/kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic test
* 话题描述： bin/kafka-topics.sh --describe --zookeeper localhost:2181 --topic test
* 删除topic： bin/kafka-topics.sh --zookeeper 127.0.0.1:2181 --delete --topic test

启动Producer

./bin/kafka-console-producer.sh --broker-list localhost:9092 --topic test

启动Consumer

0.10.0版本：/bin/kafka-console-consumer.sh --zookeeper localhost:2181 --topic test [--from-beginning]

1.1.0版本：./bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic test --from-beginning

# Git

Clone git clone https://github.com/lovemooner/moon.git

Branch

本地分支：git branch。【-a】 切换分支：git checkout branchName

Commit

git commit -a

git push origin master

git push origin HEAD:refs/for/reserved (or other branch you wish to push, start with refs/for/)（push内容在其他库并非立即可见）

撤销本地修改 git checkout filename

删除文件 git rm filename (-r 文件夹)

git log

--stat 加stat，仅显示修改的文件)

查看单个文件

git log <filename> 查看所有commit历史(包括本地未push到库里的)

--pretty=oneline pretty

-n 显示最后的几次提交日志

-p 详细包括diff内容

Config

查看当前配置 git config –list

配置：git config [--global] user.email "[email address]"

设置proxy：git config --global https.proxy http://127.0.0.1:1080

system级别 /global（用户级别）/local（当前仓库）三个 设置先从system-》global-》local

删除 git config --global --unset https.proxy

remote

git remote set-url origin [url]

-- INSERT --

## Base

[Reference](https://git-scm.com/book/zh/v2)



Workspace：工作区

Index / Stage：暂存区

Repository：仓库区（或本地仓库）

Remote：远程仓库

入门教程：[Reference](https://mp.weixin.qq.com/s?__biz=MzI2OTQ4OTQ1NQ==&mid=2247485703&idx=1&sn=69d22b19f48f6bb3cecec163f4e946eb&chksm=eadecd47dda944517f9060c59952759ad65f605adba53a2ea2b34bce1058252f799de59d9ce7&mpshare=1&scene=1&srcid=#rd])

[Reference](https://blog.csdn.net/mine_song/article/details/70770467)



全红：删除 全绿：增加

## 分支管理

## 分支查看

git branch -a 查看所有分支，包括local/remote

git branch 查看本地分支

## 分支创建

git branch mybranch 创建分支

git checkout -b branchName：相当于git branch+git checkout

分支切换





分支合并

Git merge dev：合并dev到当前分支

git rebase 和 merge 有什么区别？M(Ali)

## log

**-p 详细包括diff内容**

git log -p <filename>查看某个文件的修改历史

**-n**

显示最后的几次提交日志

git log -2

git log –p -2 查看最近2次更新内容

**show**

git show <commit-hash-id>查看某次commit的修改内容  
git show [--stat]

查看本地修改了还未提交

git diff filename

权限

查看key

cd ~/.ssh

Git VS Svn

svn更像一个文件服务器+版本控制，集中式管理文件。Git是一个分布式版本控制系统

1. 分支管理

Git擅长的是程序代码的版本化管理。

场景一:在分支开发新功能，需要新建分支修复紧急的bug

svn:创建branch--> 将代码下载到本地--> eclipse打开项目

git :创建branch-->同一工作目录下切换分支

svn下载的是完整代码库，速度慢，特别对于大型的项目，文件数达上万个以上，将非常的耗时，而git下载的是文件的镜像，大文件无需下载，而且当打开项目时需要编译，配置开发环境。像大型项目整个库的代码本地硬盘都放不下。git只下载镜像文件，需要时才下载。

2.离线提交

为了不让开发的过程代码影响其他人，svn一般task完成时一次性提交。而git在开发过程可以自由的提交，并且只有自己可见，实现本地的版本管理，非常方便。

# 第三章 分布式

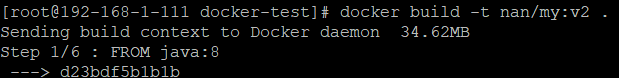
## 3.1 Docker

安装 [Reference](https://www.runoob.com/docker/centos-docker-install.html)

### 3.1.1 镜像

#### 创建镜像

docker build -t nan/my:v2 .



* --tag, -t: 镜像的名字及标签
* 最后的 . 代表在当前目录下寻找 dockerfile,也可-f指定位置。

Dockerfile

FROM java:8

VOLUME /tmp

COPY ./my-1.0-SNAPSHOT.jar my.jar

RUN bash -c "touch /my.jar"

EXPOSE 8080

ENTRYPOINT ["java","-jar","my.jar"]

~

#### 查看镜像

docker images [name]



#### docker hub

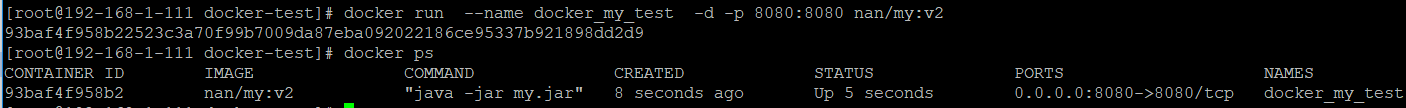
docker pull/push + name

docker search +name

### 3.1.2 容器

#### 创建容器

docker run --name names -d -p 80:80 repository:tag



* -d 后台运行
* -p 端口映射 -P容器端口随机

Docker run 将镜像放到容器中 + docker start。

启动：docker start/restart container\_id

停止：docker stop container\_id或container\_name

删除：docker rm container\_id

#### 管理容器

logs

查看内部标准输出 docker logs -f bf08b7f2cd89

管理容器 docker exec -it docker\_nginx\_v1 bash

ps

docker ps [-a]

container id image command port names

d11a0358e403 nginx:v1 "nginx -g 'daemon ..." 0.0.0.0:88->80/tcp docker\_nginx\_v2

查看容器信息

docker inspect container\_id

## 3.2 k8s

### Base

#### 基本概念

* Pod 一组关联的容器集合
* Namespace对一组资源和对象的抽象集合
* Node：Pod真正运行的主机。K8s为了管理Pod，每个Node节点上至少需要运行container runtime（Docker）、kubelet和kube-proxy服务。
* service：对一组提供相同功能的Pods的抽象
* DaemonSet 守护进程集

DaemonSet保证在特定或所有Node节点上都运行一个Pod实例，常用来部署一些集群的日志采集、监控或者其他系统管理应用。典型的应用包括:

日志收集，比如fluentd，logstash等

系统监控，比如Prometheus Node Exporter，collectd等

系统程序，比如kube-proxy, kube-dns, glusterd, ceph，ingress-controller等

* Ingress

Kubernetes中的负载均衡我们主要用到了以下两种机制：

Service：使用Service提供集群内部的负载均衡，Kube-proxy负责将service请求负载均衡到后端的Pod中

Ingress Controller：使用Ingress提供集群外部的负载均衡

#### 创建pod过程

执行kubectl run kube-nginx999 --image=nginx:latest --port=81

kubectl http--> kube-apiserver --> deployment 持久化到etcd🡪 创建ReplicaSet和Pod，每个 Pod 都被调度到合适的节点 🡪 Kubelet按PodSpec定义，创建并启动容器

#### Kubelet

集群每个 Node 节点上都启动一个Kubelet服务进程。处理 Scheduler下发到本节点任务/管理Pod生命周期。

这意味着它将处理 Pod 与 Container Runtime之间所有的转换逻辑，包括挂载卷、容器日志、垃圾回收以及其他重要事件。

可以把 Kubelet 当成一种特殊的 Controller，它每隔20秒向kube-apiserver查询 Pod，过滤 NodeName 与自身所在节点匹配的 Pod 列表。

Kubelet启动容器过程

拉取容器的镜像

通过 CRI 创建容器

启动容器

查询集群地址和凭证

kubectl config view

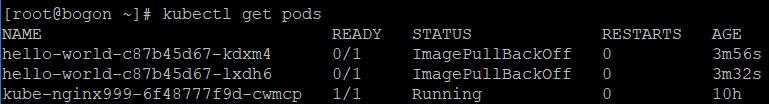
### kubectl

kubectl version

kubectl get nodes

#### 管理pods

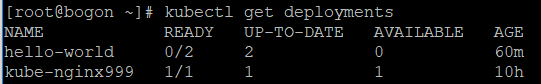
kubectl get pods [--all-namespaces]



kubectl delete pods [pod-name]

#### 管理Deployment

kubectl get deployments [name]



kubectl delete deployments name

显示 ReplicaSet 的信息

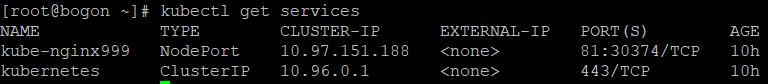
kubectl get replicasets

kubectl describe replicasets

#### 管理Service

kubectl describe services [label]

kubectl get services



kubectl delete services name

### minikube

启动：minikube start --vm-driver=none

查看状态：minikube status

minikube dashboard

kubectl proxy --port=8001 --address='192.168.0.5' --accept-hosts='^.\*' &

更新

如需更新minikube，需要更新 minikube 安装包

minikube delete 删除现有虚机，删除 ~/.minikube 目录缓存的文件

日志

minikube logs

vm driver

~/.minikube/machines/minikube/config.json

### demo

运行镜像

kubectl run kube-nginx999 --image=nginx:latest --port=81

发布服务

kubectl expose deployment kube-nginx999 --type=NodePort

#### 查看service

minikube service kube-nginx999 [–url] [-n NAMESPACE]

查看集群

kubectl cluster-info

https://192.168.0.5:8443/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy

# Java

## Jar

java-jar

-c 创建一个jar包

-t 显示jar中的内容列表

-x 解压jar包

-u 添加文件到jar包中

-f 指定jar包的文件名

-v生成详细的报造，并输出至标准设备

1）解压

Jar -xvf test.ear

2）创建

Jar -cvf test.jar ./\*

3）更新

jar -uvf RESTWebService.ear APP-INF/lib/AdfHcmUsersRestModel.jar

4）查看

jar tvf test.jar

5）搜索：

jar tvf ZhcLib.jar | grep 'pattern'

搜索整个目录下的所有jar文件：

find . -name "\*.jar"|awk '{print "jar -tvf "$1}' | sh | grep ".\*.xml"

## maven

### jar包管理策略

**1.公共依赖在的父项目中声明**

统一版本号,方便维护，尽量避免出现版本冲突。

1. **<properties>**
2. **<spring.version>**spring4.2.4**</spring.version>**
3. **<properties>**
5. **<dependencyManagement>**
6. **<dependencies>**
7. **<dependency>**
8. **<groupId>**org.springframework**</groupId>**
9. **<artifactId>**spring-beans**</artifactId>**
10. **<version>**${spring.versio}**</version>**
11. **</dependency>**
12. **</dependencies>**
13. **</dependencyManagement>**

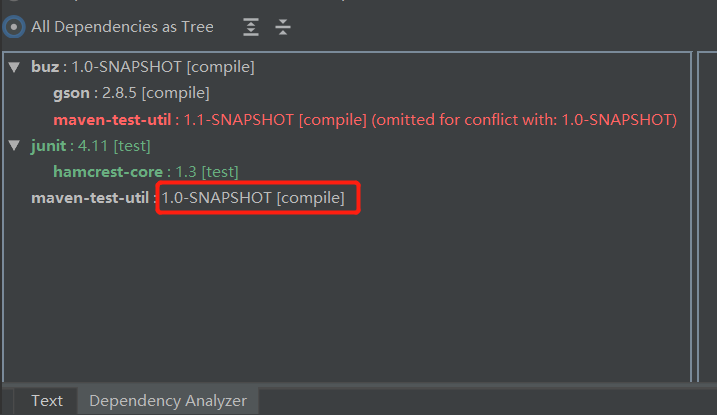
子模块使用spring-bean的jar包(不用再定义版本)：

1. **<dependencies>**
2. **<dependency>**
3. **<groupId>**org.springframework**</groupId>**
4. **<artifactId>**spring-beans**</artifactId>**
5. **</dependency>**
6. **</dependencies>**

### dependency

mvn dependency:tree [-Dverbose] >tree.txt

-Dverbose 查看jar冲突,或者用Maven helper



出现冲突原因

[Reference](https://www.cnblogs.com/ptqueen/p/7985198.html)

1.高版本兼容低版本,所以不会出现问题

2.高版本不兼容低版本。

# Other

## Zookeeper

启动

./bin/zkServer.sh start &

Kafka自带启动

./bin/zookeeper-server-start.sh ./config/zookeeper.properties

## svn

svn ls svn://slc02vlf.us.oracle.com:3690/repos

svn checkout svn://slc02vlf.us.oracle.com:3690/repos/parking

# log

svn log url

## vnc

• Kill VNC: vncserver -kill :1

• Create VNC: vncserver -geometry 1910x1050

• VNC set copy: vncconfig -nowin &

• List: vncserver -list